Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Σχολή Θετικών Επιστημών

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Εργασία στο μάθημα της Κρυπτογραφίας

Φτιάκας Σωτήριος ΑΕΜ: 3076

Μπάρμπας Γρηγόριος ΑΕΜ: 3108

10 Μαΐου 2020

Περιεχόμενα

Περίληψη

....

Θέμα 1

Θέμα 2

Θέμα 3

Θέμα 4

Θέμα 5

Dictionary Attack

```
# Import of library
from zipfile import ZipFile

# Variable names
zip_file = 'test_zip.zip'
eng_dict = 'english.txt'
check = 0

# Move the words from txt file to a list
with open(eng_dict) as ed:
    content = ed.readlines()
# you may also want to remove whitespace characters like `\n` at the end of
    each line
content = [x.strip() for x in content]

# Brute Force until we find the right password
i = 1
print('Hacking', end='')
for password in content:
    if (i%5000==1):
```

```
        print('.', end='', flush=True)
    try:
        with ZipFile(zip_file) as zf:
            zf.extractall(pwd=bytes(password,'utf-8'))
        check = 1
    except Exception:
        check = 0
        pass
    if check ==1:
        break;
    i = i + 1

print('')
print('FILE UNLOCKED')
```

Θέμα 6

Θέμα 7

Shift Operator with XOR

m: 16-bits

$$c = m \oplus (m << 6) \oplus (m << 10)$$

Όπου $m << a$ είναι κύλιση προς τα αριστερά κατά a-bits.

Για μήνυμα m και κλειδί k ισχύει: Αν $c = m \oplus k$, τότε $m = c \oplus k$

Επιπλέον, στην αρχική μας συνάρτηση κρυπτογράφησης, μπορούμε να κυλίσουμε και τα δύο μέλη ταυτόχρονα.

$$(c << 2) = (m \oplus (m << 6) \oplus (m << 10)) << 2$$

$$\Leftrightarrow (c << 2) = (m << 2) \oplus (m << 8) \oplus (m << 12)$$

Σημείωση: το $x << i$ θα συμβολίζεται ως x_i για ευκολία.
Συνεπώς θα έχουμε:

$$c_0 = m_0 \oplus m_6 \oplus m_{10} \quad (1)$$
$$c_2 = m_2 + m_8 + m_{12} \Rightarrow m_8 = m_2 + m_{12} + c_2 \quad (4)$$
$$c_4 = m_4 + m_10 + m_14 \Rightarrow m_{10} = m_4 + m_{14} + c_4 \quad (2)$$
$$c_6 = m_6 + m_{12} + m_0 \quad (5)$$
$$c_8 = m_8 + m_{14} + m_2$$
$$c_{10} = m_{10} + m_0 + m_4$$
$$c_{12} = m_{12} + m_2 + m_6$$
$$c_{14} = m_{14} + m_4 + m_8 \Rightarrow m_{14} + m_4 = m_8 + c_{14} \quad (3)$$

Ξεκινώντας από την (1) έχουμε διαδοχικά:

$$c_0 = m_0 \oplus m_6 \oplus m_{10}$$
$$(2) \Rightarrow c_0 = m_0 + m_6 + m_4 + m_{14} + c_4$$
$$(3) \Rightarrow c_0 + c_4 = m_0 + m_6 + m_8 + c_{14}$$
$$(4) \Rightarrow c_0 + c_4 + c_{14} = m_0 + m_6 + m_2 + m_{12} + c_2$$
$$(5) \Rightarrow c_0 + c_4 + c_{14} + c_2 = m_2 + c_6$$
$$\Rightarrow c_0 + c_4 + c_{14} + c_2 + c_6 = m_2 \quad (6)$$

Κάνουμε κύλιση και στα δύο μέρη του (6) προς τα δεξιά και έχουμε:

$$m_0 = c_{14} + c_2 + c_{12} + c_0 + c_4$$

και άρα τελικά έχουμε:

$$m_0 = c_0 + c_2 + c_4 + c_{12} + c_{14}$$

Κώδικας σε python

```python
import random

# Create a random message of "n" bits.
def create_m(n):
    message = []
    for i in range(n):
        bit = bool(random.getrandbits(1))
        if bit == True:
            message.append(1)
        else:
            message.append(0)

    return message #returns a list of bits

# Left rotation of a message by n.
def rotate(message, n):
    return message[n:] + message [:n] #returns a list of bits

# XOR function between a message and a key.
def xor(message, key):
    c = [0 for i in range(len(message))]
    for i in range(len(message)):
        temp = 0
```

```
        if (message[i] != key[i]):
            c[i] = 1

    return c #returns a list of bits

# Message Encryption
m0 = create_m(16) # original message
m6 = rotate(m0, 6)
m10 = rotate(m0, 10)

c0 = xor(m0, xor(m6,m10)) # c = m (+) (m<<6) (+) (m<<10)

print("Original Message: ")
print(m0)

print("Encrypted Message: ")
print(c0)

# Message Decryption
c2 = rotate(c0, 2)
c4 = rotate(c0, 4)
c12 = rotate(c0, 12)
c14 = rotate(c0, 14)
decr = xor(c0, xor(c2, xor(c4, xor(c12,c14)))) # d = c (+) (c<<2) (+) (c<<4)
    (+) (c<<12) (+) (c<<14)


print("Decrypted Message: ")
print(decr)

print("Original Message: ")
print(m0)
```

Θέμα 8

Θέμα 9

Entropy

Y/X	**0**	**1**	**2**
0	1/7	1/7	1/7
1	0	1/7	1/7
2	2/7	0	0

Αρχικά υπολογίζουμε

$$p_x(X=0) = \sum_y p_{X,Y}(0,y) = \frac{3}{7}$$
$$p_x(X=1) = \sum_y p_{X,Y}(1,y) = \frac{2}{7}$$
$$p_x(X=2) = \sum_y p_{X,Y}(2,y) = \frac{2}{7}$$
$$p_y(Y=0) = \sum_y p_{X,Y}(x,0) = \frac{3}{7}$$
$$p_y(Y=1) = \sum_y p_{X,Y}(x,1) = \frac{2}{7}$$
$$p_y(Y=2) = \sum_y p_{X,Y}(x,2) = \frac{2}{7}$$

Ισχύει ότι:

$$H(X) = -\sum_x p_X(x)\log_2 p_X(x)$$

Επομένως

$$
\begin{aligned}
H(X) &= -\frac{3}{7}\log_2\frac{3}{7} - \frac{2}{7}\log_2\frac{2}{7} - \frac{2}{7}\log_2\frac{2}{7}\\
&= -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{2}{7}\\
&\simeq 1.5566567074628228
\end{aligned}
$$

$$
\begin{aligned}
\Upsilon(Q) &= -\frac{3}{7}\log_2\frac{3}{7} - \frac{2}{7}\log_2\frac{2}{7} - \frac{2}{7}\log_2\frac{2}{7}\\
&= -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{2}{7}\\
&\simeq 1.5566567074628228
\end{aligned}
$$

Επίσης έχουμε τον εξής τύπο

$$
H(X,Y) = -\sum_x\sum_y p(x,y)\log_2 p(x,y)
$$

Άρα θα έχουμε:

$$
\begin{gathered}
H(X,\Upsilon) = -p(0,0)\log_2 p(0,0) - p(0,1)\log_2 p(0,1) - p(0,2)\log_2 p(0,2) -\\
p(1,0)\log_2 p(1,0) - p(1,1)\log_2 p(1,1) - p(1,2)\log_2 p(1,2) -\\
p(2,0)\log_2 p(2,0) - p(2,1)\log_2 p(2,1) - p(2,2)\log_2 p(2,2)\\
\simeq 2.5216406363433186
\end{gathered}
$$

Θα υπολογίσουμε την $H(Y|X)$. Χρειαζόμαστε αρχικά τα παρακάτω,

$$p_{Y|X}(y=0|x=0) = \frac{p_{X,Y}(0,0)}{p_X(0)} = \frac{\frac{1}{7}}{\frac{3}{7}} = \frac{1}{3}$$
$$p_{Y|X}(y=1|x=0) = \frac{p_{X,Y}(0,1)}{p_X(0)} = \frac{0}{\frac{3}{7}} = 0$$
$$p_{Y|X}(y=2|x=0) = \frac{p_{X,Y}(0,2)}{p_X(0)} = \frac{\frac{2}{7}}{\frac{3}{7}} = \frac{2}{3}$$
$$p_{Y|X}(y=0|x=1) = \frac{p_{X,Y}(1,0)}{p_X(1)} = \frac{\frac{1}{7}}{\frac{2}{7}} = \frac{1}{2}$$
$$p_{Y|X}(y=1|x=1) = \frac{p_{X,Y}(1,1)}{p_X(1)} = \frac{\frac{1}{7}}{\frac{2}{7}} = \frac{1}{2}$$
$$p_{Y|X}(y=2|x=1) = \frac{p_{X,Y}(1,2)}{p_X(1)} = \frac{0}{\frac{2}{7}} = 0$$
$$p_{Y|X}(y=0|x=2) = \frac{p_{X,Y}(2,0)}{p_X(2)} = \frac{\frac{1}{7}}{\frac{2}{7}} = \frac{1}{2}$$
$$p_{Y|X}(y=1|x=2) = \frac{p_{X,Y}(2,1)}{p_X(2)} = \frac{\frac{1}{7}}{\frac{2}{7}} = \frac{1}{2}$$
$$p_{Y|X}(y=2|x=2) = \frac{p_{X,Y}(2,2)}{p_X(2)} = \frac{0}{\frac{2}{7}} = 0$$

Τώρα πρέπει να υπολογίσουμε τα παρακάτω:

$$
\begin{aligned}
H(Y|X=0) &= -\sum_y p_{Y|X}(y|x=0)\log_2 p_{Y|X}(y|x=0) \\
&= -(\frac{1}{3}\log_2\frac{1}{3} + 0 + \frac{2}{3}\log_2\frac{2}{3}) \\
&= -(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}) \\
H(Y|X=1) &= -\sum_y p_{Y|X}(y|x=1)\log_2 p_{Y|X}(y|x=1) \\
&= -(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2} + 0) \\
&= -\log_2 2 = 1 \\
H(Y|X=2) &= -\sum_y p_{Y|X}(y|x=2)\log_2 p_{Y|X}(y|x=2) \\
&= -\log_2 2 = 1
\end{aligned}
$$

Τότε θα έχουμε

$$
\begin{aligned}
H(Y|X) &= \sum_x p_X x H(Y|X=x) \\
&= p_X(0)H(Y|X=0) + p_X(1)H(Y|X=1) + p_x(2)H(Y|X=2) \\
&\simeq 0.9649839288804954
\end{aligned}
$$

Γνωρίζουμε επίσης ότι από το θεώρημα της αμοιβαίας πληροφορίας έχουμε

$$
I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)
$$

Άρα έχουμε

$$H(X|Y) = -(H(Y) - H(Y|X) - H(X))$$
$$\simeq 0.9649839288804954$$

Τέλος,

$$\rho = 1 - \frac{H(Y|X)}{H(X)}$$
$$\simeq 0.5916727785823274$$

Κώδικας σε python

```
import numpy as np
import math

def create_table(elements):
    return np.reshape(elements, (int(math.sqrt(len(elements))),int(math.sqrt(
   len(elements)))))

def entropy(table):
    # Calculate pX and pY
    pX = [sum(i) for i in zip(*table)] #add columns
    pY =[sum(i) for i in table] #add rows

    # Calculate HX and HY
    HX = -sum([x*math.log(x,2) for x in pX])
    HY = -sum([x*math.log(x,2) for x in pY])

    return HX, HY, pX, pY

def mutual_entropy(table):
    temp = np.reshape(table, (len(table)**2)) # flatten table (easier to
```

```
iterate)
    temp = [i for i in temp if i != 0] # remove 0 from list (log(0) is undefined)

    HX_Y = -sum([x*math.log(x,2) for x in temp])

    return HX_Y

def committed_entropy(table):
    HX, HY, pX, pY = entropy(table)

    temp= []
    for i in range(len(table)):
        for j in range(len(table)):
            temp.append(table[j][i] / pX[i])

    temp = [i if i != 0 else 1 for i in temp] # replace 0 from list (log(0) is undefined)
                                             #log(1) = 0 so it will not affect our results

    pY_X = create_table(temp)

    HY_X_table = []
    for row in pY_X:
        HY_X_table.append(-sum([x*math.log(x,2) for x in row]))

    HY_X = sum([pX[i] * HY_X_table[i] for i in range(len(pX))])

    # Mutual Information: I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)
    HX_Y = -(HY - HY_X - HX)

    return HX_Y, HY_X
```

```python

def mutual_information(table):
    HX, _, _, _ = entropy(table)
    HX_Y, _ = committed_entropy(table)

    IX_Y = HX - HX_Y

    return IX_Y

elements = [1/7,1/7,1/7,0,1/7,1/7,2/7,0,0]
table = create_table(elements)


HX, HY, _, _ = entropy(table)
print("Entropy: ")
print(HX, HY)
print()
print("Mutual Entropy: ")
print(mutual_entropy(table))
print()
print("Committed Entropy: ")
print(committed_entropy(table))
print()
print("Mutual Information: ")
print(mutual_information(table))
```

Θέμα 10

Θέμα 11

Θέμα 12

Θέμα 13

Chinese Theorem

Έχουμε το σύστημα των γραμμικών ισοδυναμιών

$$x \equiv 9 \pmod{19}$$
$$x \equiv 9 \pmod{12}$$
$$x \equiv 13 \pmod{19}$$

Έχουμε ότι ισχύει $\gcd(12, 17, 19) = 1$, άρα δεν απαιτείται κάποια απλοποίηση.

Για την επίλυση του συστήματος χρησιμοποιούμε το Κινέζικο Θεώρημα Υπολοίπων

Έτσι έχουμε: $m = 17 * 12 * 19 = 3876$

$$\begin{aligned} M_1 &= 228y_1 \equiv 1 \mod 17 \implies 7y_1 \equiv \mod 17 \implies y_1 = 5 \\ M_2 &= 323y_2 \equiv 1 \mod 12 \implies 11y_1 \equiv \mod 12 \implies y_1 = 11 \\ M_3 &= 204y_3 \equiv 1 \mod 19 \implies 14y_1 \equiv \mod 19 \implies y_1 = 15 \end{aligned}$$

Τώρα πολλαπλασιάζουμε και προσθέτουμε:

$$\begin{aligned} x &= 9 * 228 * 5 + 9 * 323 * 11 + 13 * 204 * 15 \\ &= 82017(1) \end{aligned}$$

Παρατηρούμε ότι η (1) γράφεται,

$$x = 82017 = 621 + 3876k, k \in \mathbb{Z}$$

Για $k = 0$ έχουμε λύση το $x = 621$

Κώδικας σε python

```
# Brute force algorith to test results
for x in range(1,1000):
    if ((x % 17) == 9) and ((x % 12) == 9) and ((x % 19) == 13):
        print(x)
```

Θέμα 14

Θέμα 15

Θέμα 16

Θέμα 17

Θέμα 18

Θέμα 19

Θέμα 20

Θέμα 21

Θέμα 22

Αρχικά θέλουμε να αποδείξουμε ότι οι αριθμοί της μορφής $4n$+3 δεν είναι τέλεια τετράγωνα. Αρχικά για $n \leq 0$ εύκολα παρατηρούμε ότι ισχύει η παραπάνω πρόταση. Αυτό συμβαίνει καθώς για $n = 0$ έχουμε το 3 το οποίο δεν είναι τέλειο τετράγωνο και για $n < 0$ το $4n + 3$ είναι αρνητικός.

Έστω ότι,

$$4n + 3 = a^2, \quad n, a \in \mathbb{N}^*(1)$$

Εφόσον $a \in \mathbb{N}^*$, τότε μπορούμε να πούμε ότι $a = 2k + 1, \quad k \in \mathbb{N}$.

Αντικαθιστώντας το a στην (1) έχουμε:

$$\begin{aligned} 4n + 3 = (2k + 1)^2 \implies 4n + 3 &= 4k^2 + 4k + 1 \\ &\equiv 4k^2 + 4k - 4n = 2 \\ &\equiv 2(k^2 + k - n) = 1 \\ &\equiv k^2 + k - n = \frac{1}{2} \end{aligned}$$

Το οποίο είναι άτοπο καθώς $k, n \in \mathbb{N}^*$ και άρα το $k^2 \in \mathbb{N}^*$ αλλά και όλη η παράσταση $k^2 + k - n \in \mathbb{N}$, εφόσον είναι άθροισμα των φυσικών αριθμών k^2, k και $-n$.

Συνεπώς και η αρχική ισοδύναμη υπόθεση είναι άτοπη, οπότε το $4n+3$ δεν είναι τέλειο τετράγωνο.

Για το δεύτερο ερώτημα παρατηρούμε ότι όλοι αριθμοί

$$11, 111, \ldots, 111 \cdots 111, \ldots$$

μπορούν να γραφούν στην μορφής $(4n + 3) + 10^q$, συνεπώς αυτό το σύνολο αριθμών δεν θα έχει τέλειο τετράγωνο.

Θέμα 23

Θέμα 24

Θέμα 25

Θέμα 26

Θέμα 27

Θέμα 28

Θέμα 29

Θέμα 30

Υπόθεση:

$N > 2$

$N = p_1 p_2 \ldots p_k$

$p_j - 1 | N - 1 \forall j$

Απόδειξη:

Έστω $\gcd(a, N) = 1$.

Από το θεώρημα του Fermat, $\forall j$, έχουμε $a^{p_j} \equiv 1 \mod p_j$.

Εφόσον $p_j - 1 | N - 1$,και άρα $a^{N-1} \equiv 1 \mod p_j$.

Δηλαδή το $a^{N-1} - 1$ είναι πολλαπλάσιο κάθε p_j.

Συνεπώς $a^{N-1} \equiv 1 \mod N$.

Θέμα 31

Αναφορές